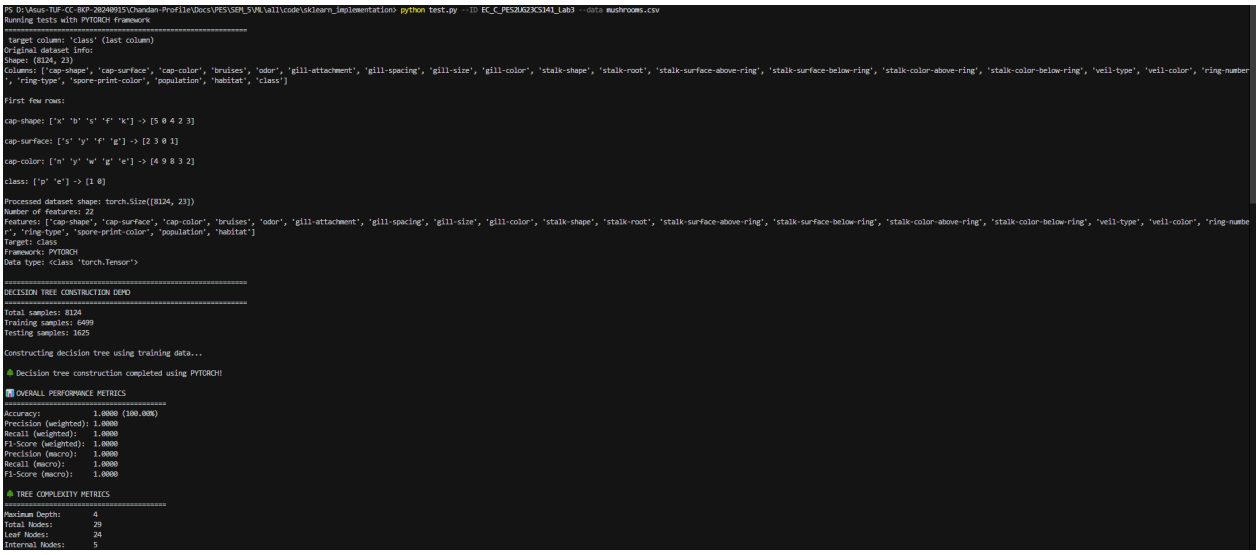
ML Lab-2

Week-2

Name: Chandan Chatragadda	SRN: PES2UG23CS141	Section: 5C

Week: 2

Mushrooms.csv

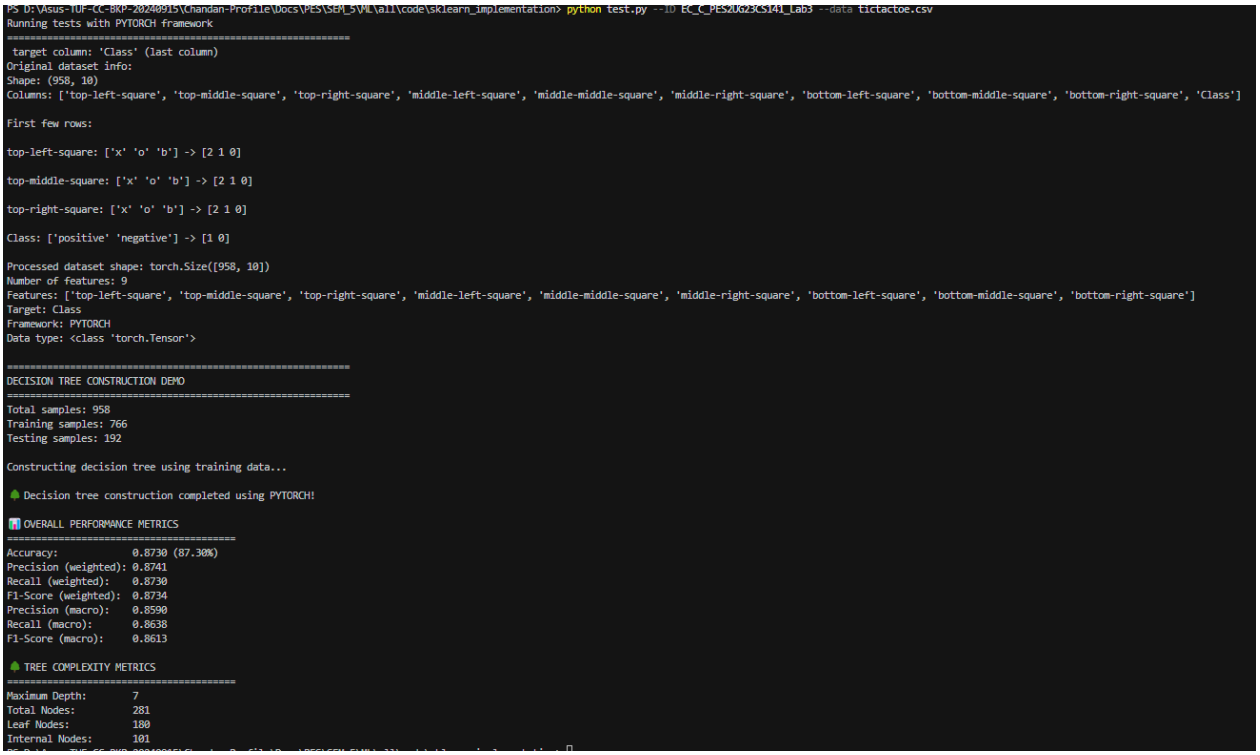


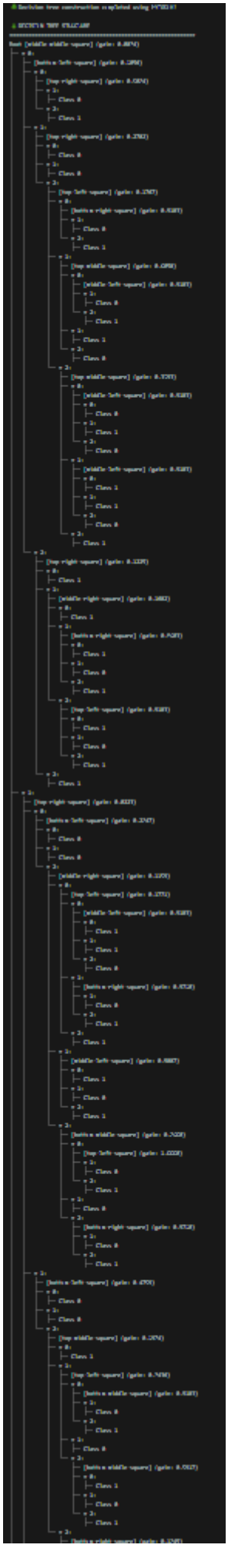
```
🌲 DECISION TREE STRUCTURE
============================================================
Root [odor] (gain: 0.9083)
├── = 0:
│   ├── Class 0
├── = 1:
│   ├── Class 1
├── = 2:
│   ├── Class 1
├── = 3:
│   ├── Class 0
├── = 4:
│   ├── Class 1
├── = 5:
│   ├── [spore-print-color] (gain: 0.1469)
│   ├── = 0:
│   │   ├── Class 0
│   ├── = 1:
│   │   ├── Class 0
│   ├── = 2:
│   │   ├── Class 0
│   ├── = 3:
│   │   ├── Class 0
│   ├── = 4:
│   │   ├── Class 0
│   ├── = 5:
│   │   ├── Class 1
│   ├── = 7:
│   │   ├── [habitat] (gain: 0.2217)
│   │   ├── = 0:
│   │   │   ├── [gill-size] (gain: 0.7642)
│   │   │   ├── = 0:
│   │   │   │   ├── Class 0
│   │   │   └── = 1:
│   │   │       ├── Class 1
│   │   ├── = 1:
│   │   │   ├── Class 0
│   │   ├── = 2:
│   │   │   ├── [cap-color] (gain: 0.7300)
│   │   │   ├── = 1:
│   │   │   │   ├── Class 0
│   │   │   ├── = 4:
│   │   │   │   ├── Class 0
│   │   │   ├── = 8:
│   │   │   │   ├── Class 1
│   │   │   └── = 9:
│   │   │       ├── Class 1
│   │   ├── = 4:
│   │   │   ├── Class 0
│   │   └── = 6:
│   │       ├── Class 0
│   └── = 8:
│       ├── Class 0
├── = 6:
│   ├── Class 1
├── = 7:
│   ├── Class 1
├── = 8:
│   ├── Class 1
```

Nursery.csv

```
PS D:\Asus-TUF-CC-BKP-20240915\Chandan-Profile\Docs\PES\SEM_5\ML\all\code\sklearn_implementation> python test.py --ID EC_C_PES2UG23CS141_Lab3 --data Nursery.csv
Running tests with PYTORCH framework
============================================================
 target column: 'class' (last column)
Original dataset info:
Shape: (12960, 9)
Columns: ['parents', 'has_nurs', 'form', 'children', 'housing', 'finance', 'social', 'health', 'class']

First few rows:

parents: ['usual' 'pretentious' 'great_pret'] -> [2 1 0]

has_nurs: ['proper' 'less_proper' 'improper' 'critical' 'very_crit'] -> [3 2 1 0 4]

form: ['complete' 'completed' 'incomplete' 'foster'] -> [0 1 3 2]

class: ['recommend' 'priority' 'not_recom' 'very_recom' 'spec_prior'] -> [2 1 0 4 3]

Processed dataset shape: torch.Size([12960, 9])
Number of features: 8
Features: ['parents', 'has_nurs', 'form', 'children', 'housing', 'finance', 'social', 'health']
Target: class
Framework: PYTORCH
Data type: <class 'torch.Tensor'>

============================================================
DECISION TREE CONSTRUCTION DEMO
============================================================
Total samples: 12960
Training samples: 10368
Testing samples: 2592

Constructing decision tree using training data...

🌳 Decision tree construction completed using PYTORCH!

📊 OVERALL PERFORMANCE METRICS
====================================
Accuracy:            0.9867 (98.67%)
Precision (weighted): 0.9876
Recall (weighted):    0.9867
F1-Score (weighted):  0.9872
Precision (macro):    0.7604
Recall (macro):       0.7654
F1-Score (macro):     0.7628

🌳 TREE COMPLEXITY METRICS
========================================
Maximum Depth:       7
Total Nodes:         952
Leaf Nodes:          680
Internal Nodes:      272
```

Tictactoe.csv:




Decision tree construction completed using 'gini'
DECISION TREE STRUCTURE
Root [middle middle-square] (gain: 0.0874)
n 0:
[bottom left-square] (gain: 0.1294)
n 0:
[top right-square] (gain: 0.0263)
n 1:
Class 0
n 2:
Class 1
n 1:
[top right-square] (gain: 0.2762)
n 0:
Class 0
n 1:
Class 0
n 2:
[top left-square] (gain: 0.1747)
n 0:
[bottom right-square] (gain: 0.0187)
n 1:
Class 0
n 2:
Class 1
n 1:
[top middle-square] (gain: 0.0958)
n 0:
[middle left-square] (gain: 0.0187)
n 1:
Class 0
n 2:
Class 1
n 1:
Class 1
n 2:
Class 0
n 2:
[top middle-square] (gain: 0.3393)
n 0:
[middle left-square] (gain: 0.0187)
n 0:
Class 0
n 1:
Class 1
n 2:
Class 0
n 1:
[middle left-square] (gain: 0.0187)
n 0:
Class 1
n 1:
Class 1
n 2:
Class 0
n 2:
Class 1
n 2:
[top right-square] (gain: 0.1329)
n 0:
Class 1
n 1:
[middle right-square] (gain: 0.1882)
n 0:
Class 1
n 1:
[bottom right-square] (gain: 0.0187)
n 0:
Class 1
n 1:
Class 0
n 2:
Class 1
n 2:
[top left-square] (gain: 0.0187)
n 0:
Class 1
n 1:
Class 0
n 2:
Class 1
n 2:
Class 1
n 1:
[top right-square] (gain: 0.0333)
n 0:
[bottom left-square] (gain: 0.2747)
n 0:
Class 0
n 1:
Class 0
n 2:
[middle right-square] (gain: 0.1370)
n 0:
[top left-square] (gain: 0.2752)
n 0:
[middle left-square] (gain: 0.0187)
n 0:
Class 1
n 1:
Class 1
n 2:
Class 0
n 1:
[bottom right-square] (gain: 0.0728)
n 0:
Class 0
n 1:
Class 1
n 2:
Class 1
n 1:
[middle left-square] (gain: 0.0667)
n 0:
Class 1
n 1:
Class 0
n 2:
Class 1
n 2:
[bottom middle-square] (gain: 0.7220)
n 0:
[top left-square] (gain: 1.0000)
n 1:
Class 0
n 2:
Class 1
n 1:
Class 0
n 2:
[bottom right-square] (gain: 0.0728)
n 1:
Class 0
n 2:
Class 1
n 1:
[bottom left-square] (gain: 0.4703)
n 0:
Class 0
n 1:
Class 0
n 2:
[top middle-square] (gain: 0.1874)
n 0:
Class 1
n 1:
[top left-square] (gain: 0.7438)
n 0:
[bottom middle-square] (gain: 0.0187)
n 1:
Class 0
n 2:
Class 1
n 1:
Class 0
n 2:
[bottom middle-square] (gain: 0.0937)
n 0:
Class 1
n 1:
Class 0
n 2:
Class 1
n 2:

a) Algorithm Performance

Qn. Which dataset achieved the highest accuracy and why?

Ans: Mushroom dataset. As it has a properly defined dataset.

Qn. How does dataset size affect performance?

Ans: Bigger datasets have more information, boundaries and improve accuracy.

Qn. What role does the number of features play?

Ans: More features can make the model learn complex patterns and give better predictions and perform better in the real world datasets.

b) Data Characteristics Impact

Qn. How does class imbalance affect tree construction?

Ans: Class imbalance can bias the tree towards majority classes and give partial results.

Qn. Which types of features (binary vs multi-valued) work better?

Ans: Binary features often create simpler and clearer splits. Multi-valued categorical features offer richer information but

increases tree complexity.

c) Practical Applications

Qn. For which real-world scenarios is each dataset type most relevant?

Ans:

● Mushroom: Food safety and scientific studies.

● Tic-tac-toe: Game outcome prediction or training a game bot.

● Nursery: Social and family-based recommendation systems with multiple categories reflecting real-world complexity.

Qn. What are the interpretability advantages for each domain?

Ans: Mushroom dataset's binary classification and features give direct outcome of whether it is safe or poisonous.

Tictactoe dataset's game states map intuitively to tree splits and helps in understanding.

The nursery dataset, with multi-class and more complex categorical factors, offers richer but more complex interpretability challenges.

d) How to Improve Performance for Each Dataset

● Mushroom: Reduce overfitting, add feature selection and prune unnecessary features.

● Tic-tac-toe: Tune tree depth to prevent overfitting.

● Nursery: Handle class imbalance and consider hierarchical tree structures due to multi-class target and complex features.